**6A]** Write an assembly language program to calculate the factorial of the entered number using

procedures.

**Program:**

section .data

read db "Enter a number: "

readlen equ $-read

write db "The factorial is: "

writelen equ $-write

newline db '', 10

nl equ $-newline

section .bss

num resb 9

inter resb 9

prod resb 9

fact resb 9

count resb 9

section .text

global \_start

\_start:

; read number

mov eax, 4

mov ebx, 1

mov ecx, read

mov edx, readlen

int 80h

mov eax, 3

mov ebx, 2

mov ecx, num

mov edx, 9

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

; Display statement

mov eax, 4

mov ebx, 1

mov ecx, write

mov edx, writelen

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

; assign 1 to PROD

mov eax, '1'

mov [prod], eax

; assign 'num' to 'inter'

mov eax, [num]

mov [inter], eax

; assign 'num' to 'count'

mov eax, [num]

mov [count], eax

; Procedure call

call Factorial

jmp L3

L3:

; print Factorial

mov eax, 4

mov ebx, 1

mov ecx, prod

mov edx, 5

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

; exit

mov eax, 1

mov ebx, 0

int 80h

Factorial:

jmp L1

L1:

; Mulitply

mov al, [prod]

sub al, '0'

mov bl, [inter]

sub bl, '0'

mul bl

add al, '0'

mov [prod], al

; decrement 'inter'

mov eax, [inter]

dec eax

mov [inter], eax

jmp L2

L2:

; Compare to run loop

mov al, [inter]

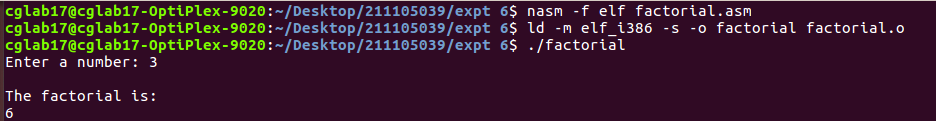
mov bl, '1'

cmp al, bl

jge L1

ret

**Output:**



**6B]** Write an assembly language program to display Fibonacci series up to n using procedures.

**Program:**

section .data

string1 db "Enter size of fibo series: ", 10

string1len equ $-string1

string2 db "The series is: ", 10

string2len equ $-string2

newline db '', 10

nl equ $-newline

empty db "-"

emptylen equ $-empty

section .bss

num resb 5

a resb 5

b resb 5

c resb 5

inter resb 5

count resb 5

section .text

global \_start

\_start:

mov eax, 4

mov ebx, 1

mov ecx, string1

mov edx, string1len

int 80h

; [num] stores the size of fibo

mov eax, 3

mov ebx, 2

mov ecx, num

mov edx, 5

int 80h

; To print the series

mov eax, 4

mov ebx, 1

mov ecx, string2

mov edx, string2len

int 80h

; Compare the size of Fibo entered

mov al, [num]

mov bl, '0'

cmp al, bl

je Empty

jg Compare

Empty:

; if (num == 0) print '-'

mov eax, 4

mov ebx, 1

mov ecx, empty

mov edx, emptylen

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

; exit

jmp C3

Compare:

; Compare if (num >= 1)

mov al, [num]

mov bl, '1'

cmp al, bl

je PrintOnce

jg PrintTwo

PrintOnce:

; if (num == 1)

mov eax, '0'

mov [a], eax

mov eax, 4

mov ebx, 1

mov ecx, a

mov edx, 5

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

jmp C3

PrintTwo:

; if (num >= 2)

; Initialize 'a' and 'b'

mov eax, '0'

mov [a], eax

mov eax, '1'

mov [b], eax

mov eax, 4

mov ebx, 1

mov ecx, a

mov edx, 5

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

mov eax, 4

mov ebx, 1

mov ecx, b

mov edx, 5

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

jmp C1

C1:

; Compare if (num > 2), and print rest series

mov al, [num]

mov bl, '2'

cmp al, bl

je C3

jg C2

C2:

; Initialize count

mov eax, '2'

mov [count], eax

call fibo

jmp C3

C3:

; exit program

mov eax, 1

mov ebx, 0

int 80h

fibo:

jmp L1

L1:

;--------- addition a, b, c

mov eax, [a]

sub eax, '0'

mov ebx, [b]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [c], eax

; ------- print 'c'

mov eax, 4

mov ebx, 1

mov ecx, c

mov edx, 5

int 80h

mov eax, 4

mov ebx, 1

mov ecx, newline

mov edx, nl

int 80h

; ------- Move b to a

mov eax, [b]

mov [a], eax

; ------- Move c to a

mov eax, [c]

mov [b], eax

; ------ Increment count

mov eax, [count]

inc eax

mov [count], eax

jmp L2

L2:

; ------ Compare count and num

mov al, [count]

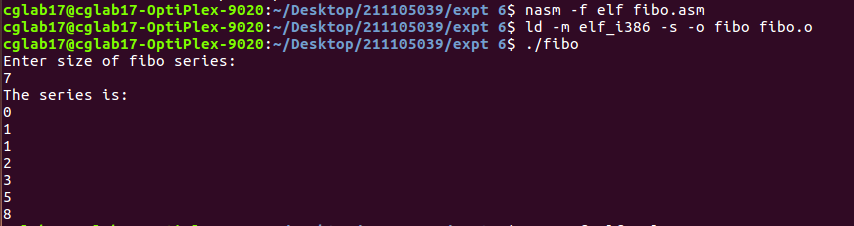
mov bl, [num]

cmp al, bl

jl L1

ret

**Output:**



**6C]** Write an assembly language program to implement a calculator program using procedures.

**Program:**

section .data

read1 db "Enter numbers: ", 10

numlen1 equ $-read1

nl db "", 10

nllen equ $-nl

printSum db "The sum ", 9

Slen equ $-printSum

printDiff db "The difference ", 9

Dlen equ $-printDiff

printProd db "The product ", 9

Plen equ $-printProd

printQuo db "The quotient ", 9

Qlen equ $-printQuo

section .bss

num1 resb 9

num2 resb 9

sum resb 9

diff resb 9

prod resb 9

quo resb 9

remind resb 9

section .text

global \_start

\_start:

mov eax, 4

mov ebx, 1

mov ecx, read1

mov edx, numlen1

int 80h

; READ NUMBER 1

mov eax, 3

mov ebx, 2

mov ecx, num1

mov edx, 9

int 80h

; READ NUMBER 2

mov eax, 3

mov ebx, 2

mov ecx, num2

mov edx, 9

int 80h

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

; addition

call Addition

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

; subtraction

call Subtraction

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

; multiplication

call Multiply

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

; division

call Divide

; exit

mov eax, 1

mov ebx, 0

int 80h

Addition:

mov eax, [num1]

mov ebx, [num2]

sub eax, '0'

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [sum], eax

; print sum

mov eax, 4

mov ebx, 1

mov ecx, printSum

mov edx, Slen

int 80h

mov eax, 4

mov ebx, 1

mov ecx, sum

mov edx, 9

int 80h

ret

Subtraction:

mov eax, [num1]

mov ebx, [num2]

sub eax, '0'

sub ebx, '0'

sub eax, ebx

add eax, '0'

mov [diff], eax

; print difference

mov eax, 4

mov ebx, 1

mov ecx, printDiff

mov edx, Dlen

int 80h

mov eax, 4

mov ebx, 1

mov ecx, diff

mov edx, 9

int 80h

ret

Multiply:

mov al, [num1]

mov bl, [num2]

sub al, '0'

sub bl, '0'

mul bl

add al, '0'

mov [prod], al

; print product

mov eax, 4

mov ebx, 1

mov ecx, printProd

mov edx, Plen

int 80h

mov eax, 4

mov ebx, 1

mov ecx, prod

mov edx, 9

int 80h

ret

Divide:

mov al, [num1]

mov bl, [num2]

sub al, '0'

sub bl, '0'

div bl

add al, '0'

mov [quo], al

; print quotient

mov eax, 4

mov ebx, 1

mov ecx, printQuo

mov edx, Qlen

int 80h

mov eax, 4

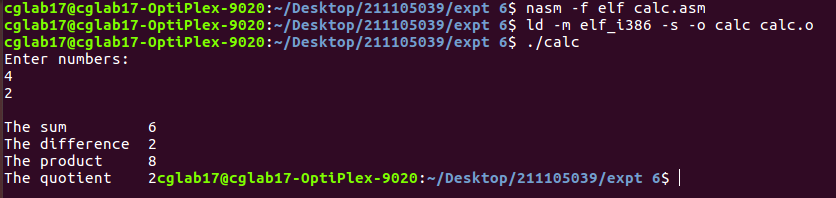
mov ebx, 1

mov ecx, quo

mov edx, 9

int 80h

ret

**Output:**